Data Science for Software Engineering:

Important Considerations and Typical Setbacks

Leandro L. Minku

University of Birmingham, UK

Research Interests

- Machine learning:
  - Machine learning for non-stationary environments.
  - Class imbalance learning.
  - Ensembles of learning machines.

- Machine learning for software engineering:
  - Software effort estimation.
  - Prediction of defect-inducing software changes.

- Search-based software engineering:
  - Software project scheduling.
  - Software architecture optimisation.

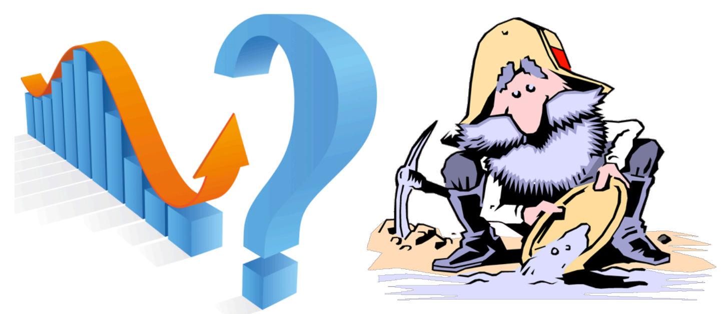
Software Engineering Data

Software engineering processes and products have been generating a wealth of data.

Bug Reports

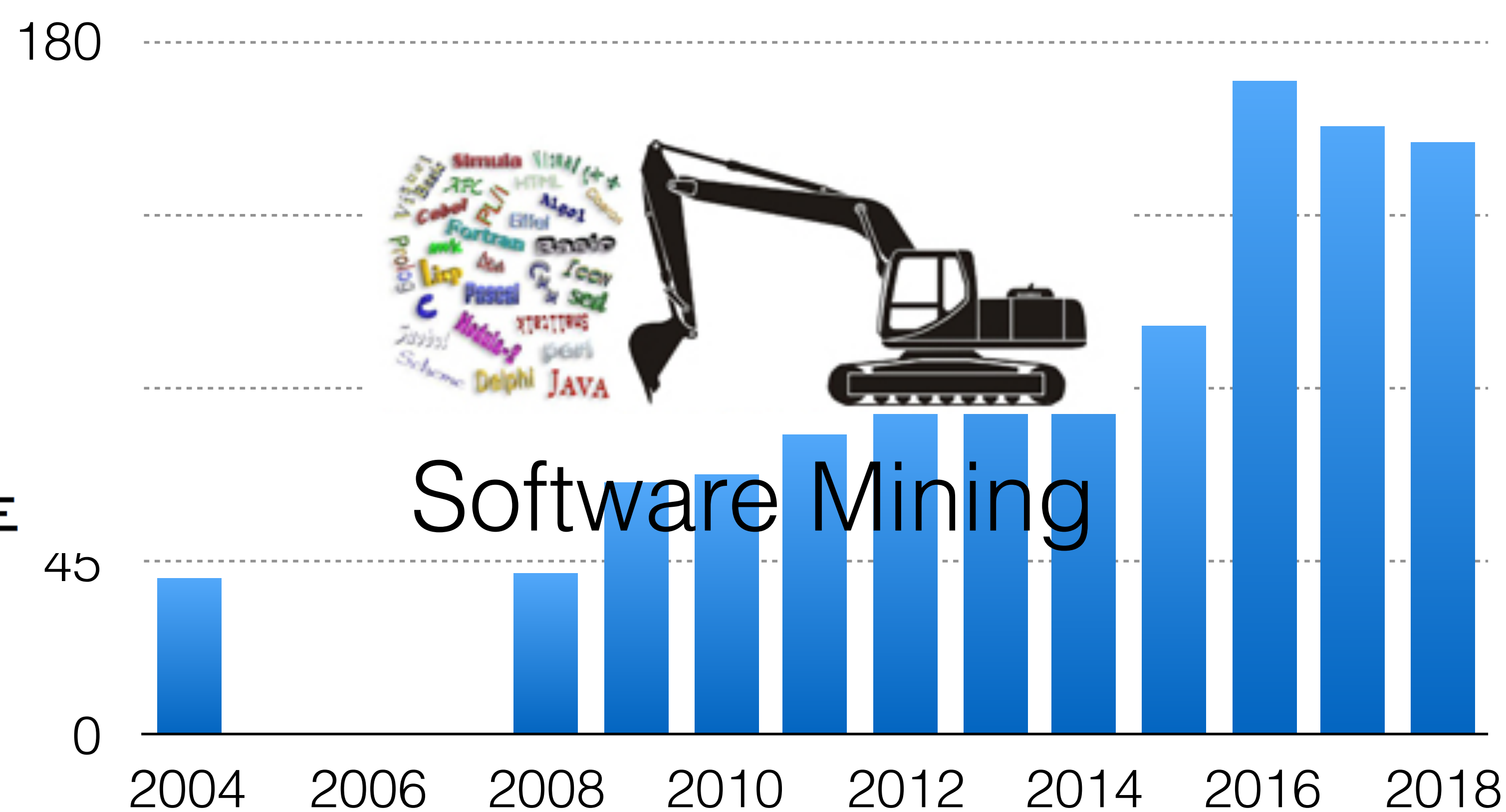
Software Crashes

Increase on Data Science for Software Engineering Research



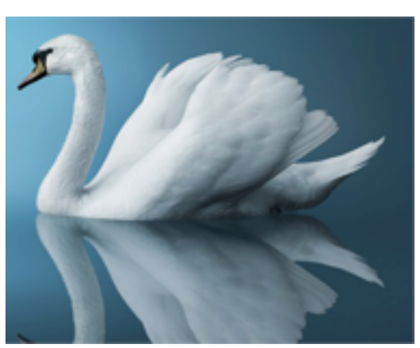
PROMISE MSR

Number of Research Paper Submissions



RAISE

Software Mining



SWAN

In this talk…

Discussion of:

- important points to consider when working with data science for software engineering, and
- typical setbacks resulting from overlooking them.

Focus: predictive analytics.

- Based on a training set **D** $\in$ **X** x **Y**, learn f: **X** —> **Y.**
- **X** are input features (a.k.a., input attributes, independent variables).
- **Y** are output features (a.k.a., output attributes, dependent variables).

Example: Software Defect Prediction

Components from previous versions

x_1 (LOC)	x_2 (Halstead)	x_3 (Cyclomatic)	…	y (defective?)
1000	80	70	…	Yes
700	30	40	…	No
800	35	30	…	No
…	…	…	…	…

→ Machine Learning Algorithm →

Predictive Model

New component **x**
of new version

x_1 (LOC)	x_2 (Halstead)	x_3 (Cyclomatic)	…
1000	80	70	…

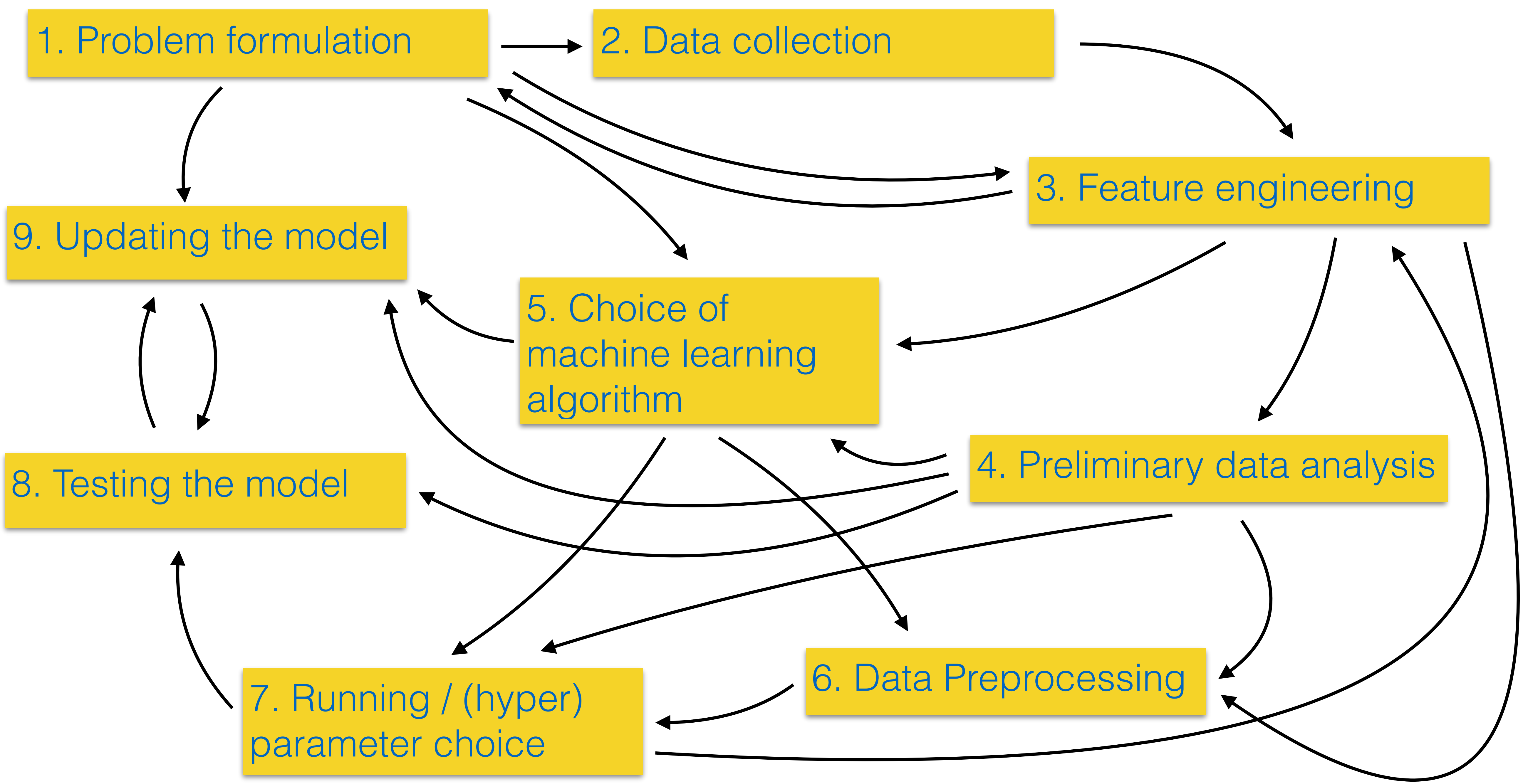
→

→ Defective?

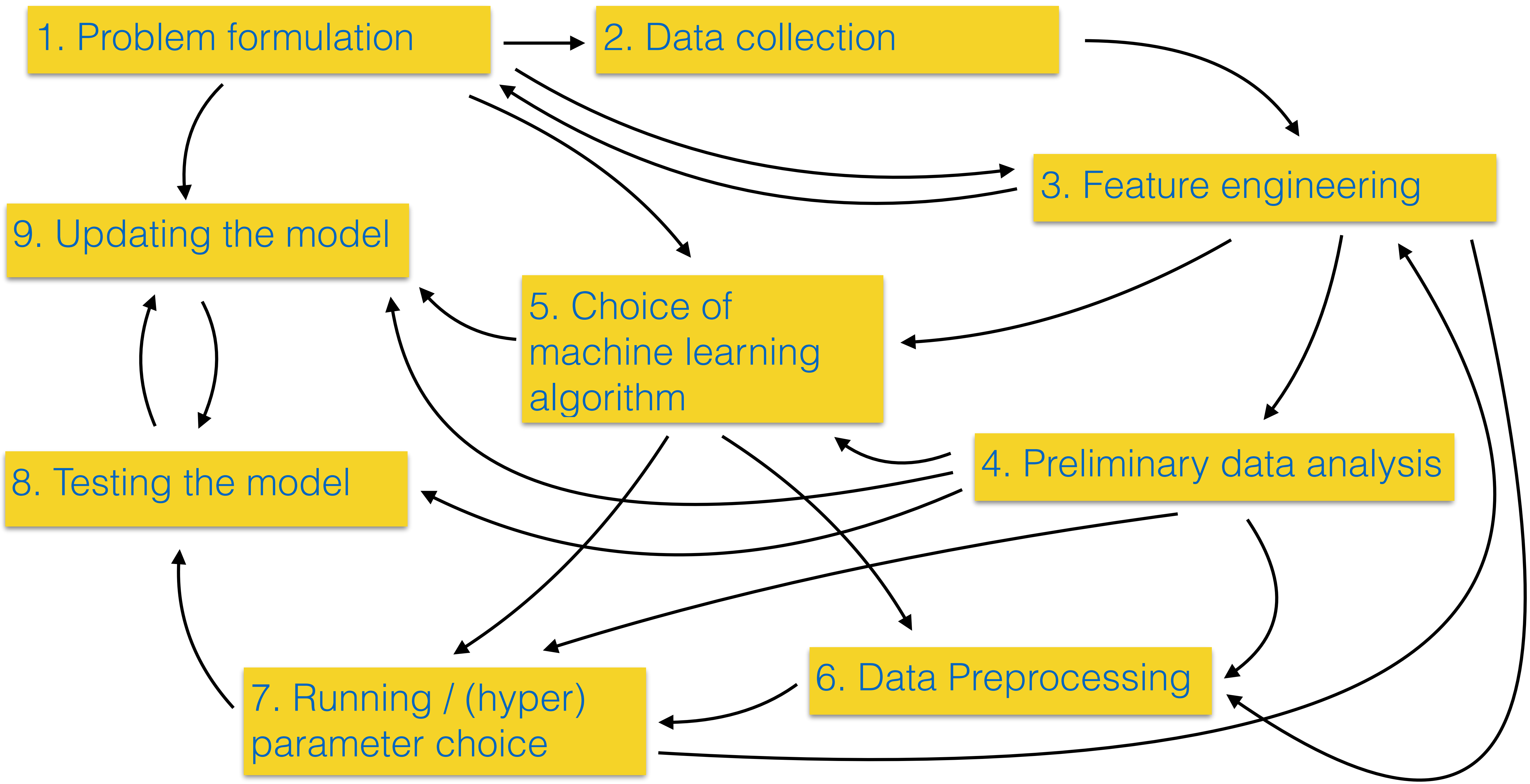
Data Science Involves Several Steps

1. Problem formulation.
2. Data collection.
3. Feature engineering.
4. Preliminary analysis of the data.
5. Choice of machine learning algorithms.
6. Data preprocessing.
7. Running / (hyper) parameter choice for machine learning algorithms.
8. Testing the selected model.
9. Updating the model.

Data Science Involves Several Interdependent Steps



Data Science Involves Several Interdependent Steps



High Level Consideration

Reflect upon each of these steps in detail!

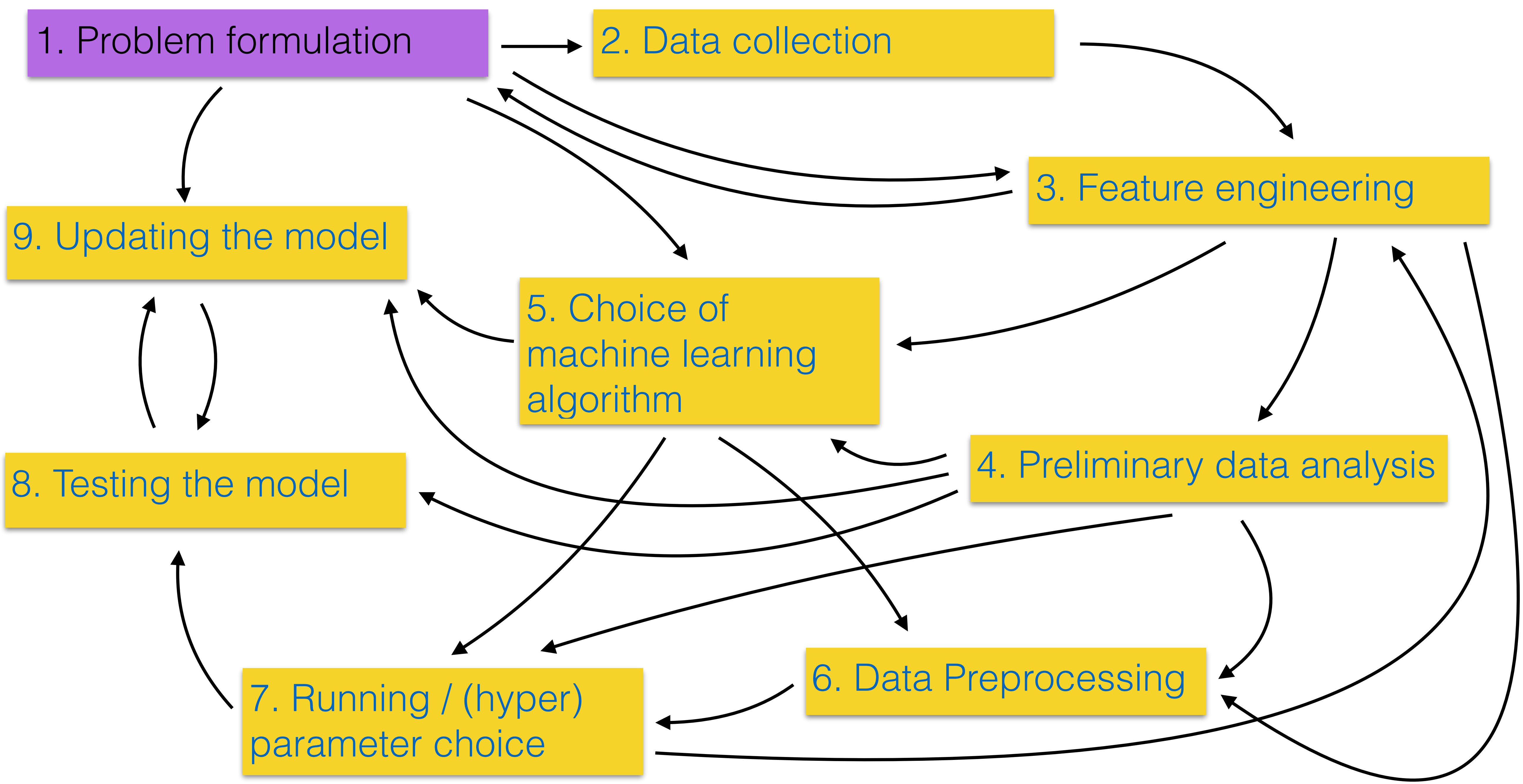
Four Considerations

- Problem relevance.
- Multi-source and temporal data.
- Class imbalance.
- Parameter tuning.

Four Considerations

- **Problem relevance.**
- Multi-source and temporal data.
- Class imbalance.
- Parameter tuning.

Data Science Involves Several Interdependent Steps



What Is The Problem?

File **x**
of a version of
the software →  → #defects

What Is The Problem?

File **x**
of a version of
the software

→

→

#defects
defective?

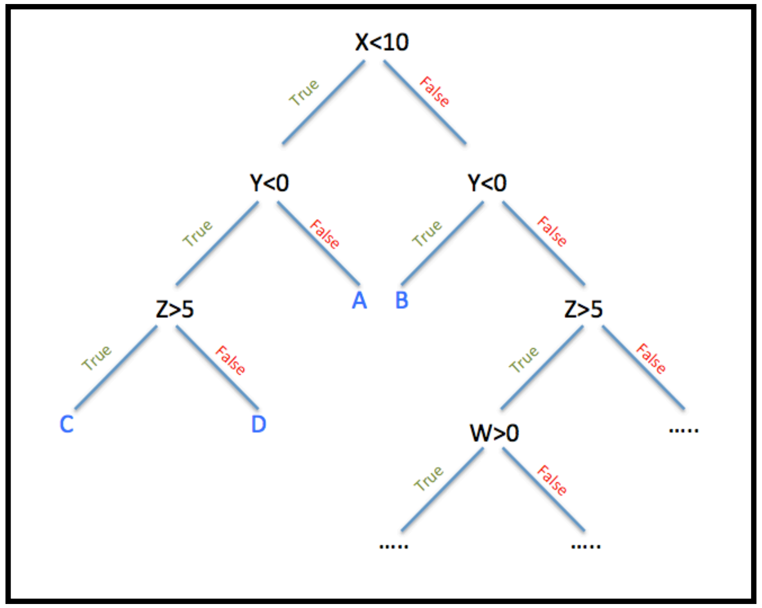
What Is The Problem?

Version of
the software x

ranking of
files based on
defect-proneness

What Is The Problem?

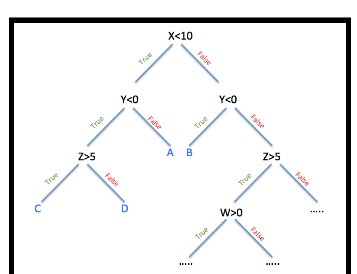
File **x**
of a version of
the software



defective?

Typical Setback

To adopt a problem that is not really useful for your **targeted** practitioners.



Related Setback

Overlook potential variations of the problem, which may be easier to solve and be equally valuable for the targeted practitioners.

ranking? #defects?

Avoiding Setback

Talk with the targeted practitioners!

Investigate alternative problem formulations.

What is the Problem?

Version of
the software x

ranking of
files based on
defect-proneness

Why Is It Relevant?

- The company develops business software.
- Several versions of the software are typically rolled out.
- Once a given version is implemented, each of its source code files passes through a testing phase in a waterfall style.
- Testing resources are limited. The company want tools to help them allocating testing resources to make testing more cost-effective.

Why Is It Relevant?

- **The company develops business software.**
  - The company can afford some software components to be more well tested than others.
- Several versions of the software are typically rolled out.
- Once a given version is implemented, each of its source code files passes through a testing phase in a waterfall style.
- Testing resources are limited. The company want tools to help them allocating testing resources to make testing more cost-effective.

Why Is It Relevant?

- The company develops business software.
- **Several versions of the software are typically rolled out.**
  - It is reasonable to use knowledge from past versions to learn how to rank.
- Once a given version is implemented, each of its source code files passes through a testing phase in a waterfall style.
- Testing resources are limited. The company want tools to help them allocating testing resources to make testing more cost-effective.

Why Is It Relevant?

- The company develops business software.
- Several versions of the software are typically rolled out.
- **Once a given version is implemented, each of its source code files passes through a testing phase in a waterfall style.**
    - Ranking is produced after the whole new version of the software is developed.
- Testing resources are limited. The company want tools to help them allocating testing resources to make testing more cost-effective.

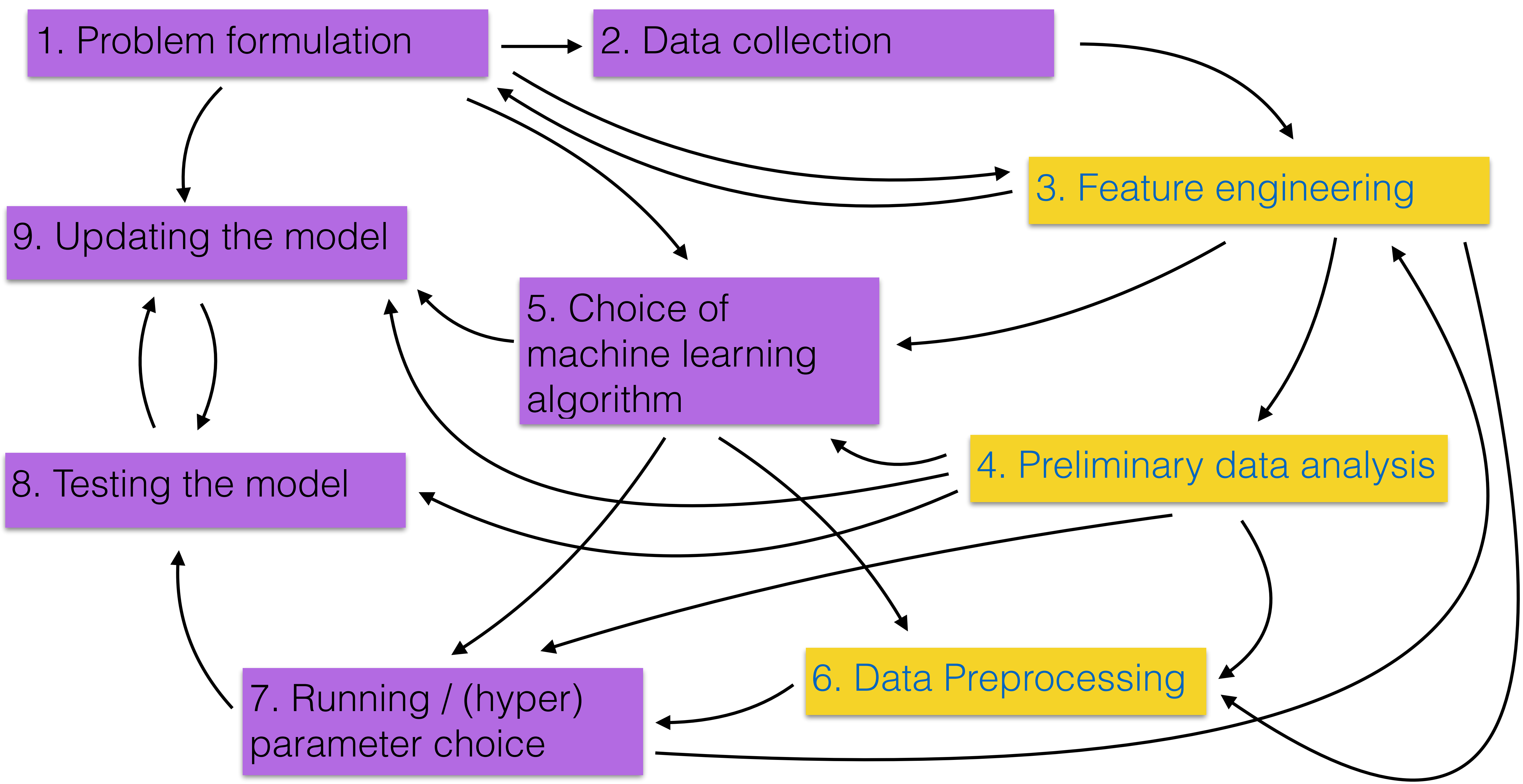
Why Is It Relevant?

- The company develops business software.
- Several versions of the software are typically rolled out.
- Once a given version is implemented, each of its source code files passes through a testing phase in a waterfall style.
- **Testing resources are limited. The company want tools to help them allocating testing resources to make testing more cost-effective.**
    - Ranking could enable to allocate more test resources to the top ranked files, until the resources (almost) finish.
    - The company favours predictive performance over model readability.

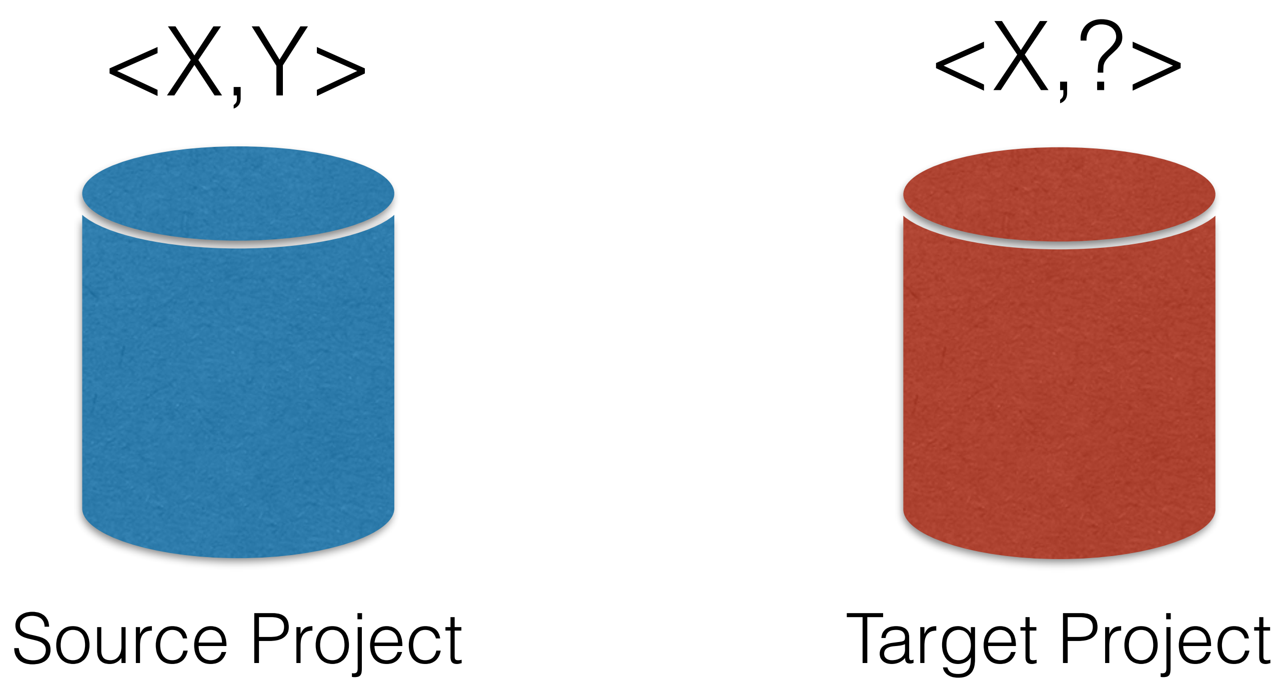
Four Considerations

- Problem formulation.
- **Multi-source and temporal data.**
- Class imbalance.
- Parameter tuning.

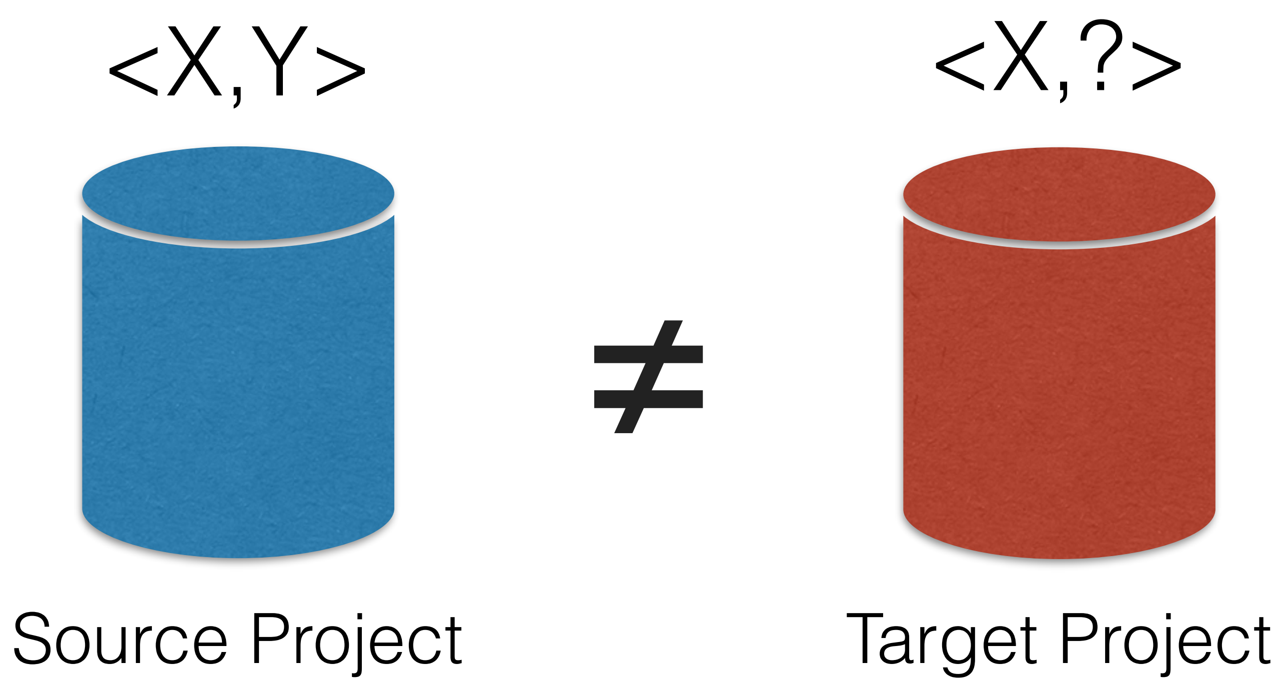
Data Science Involves Several Interdependent Steps



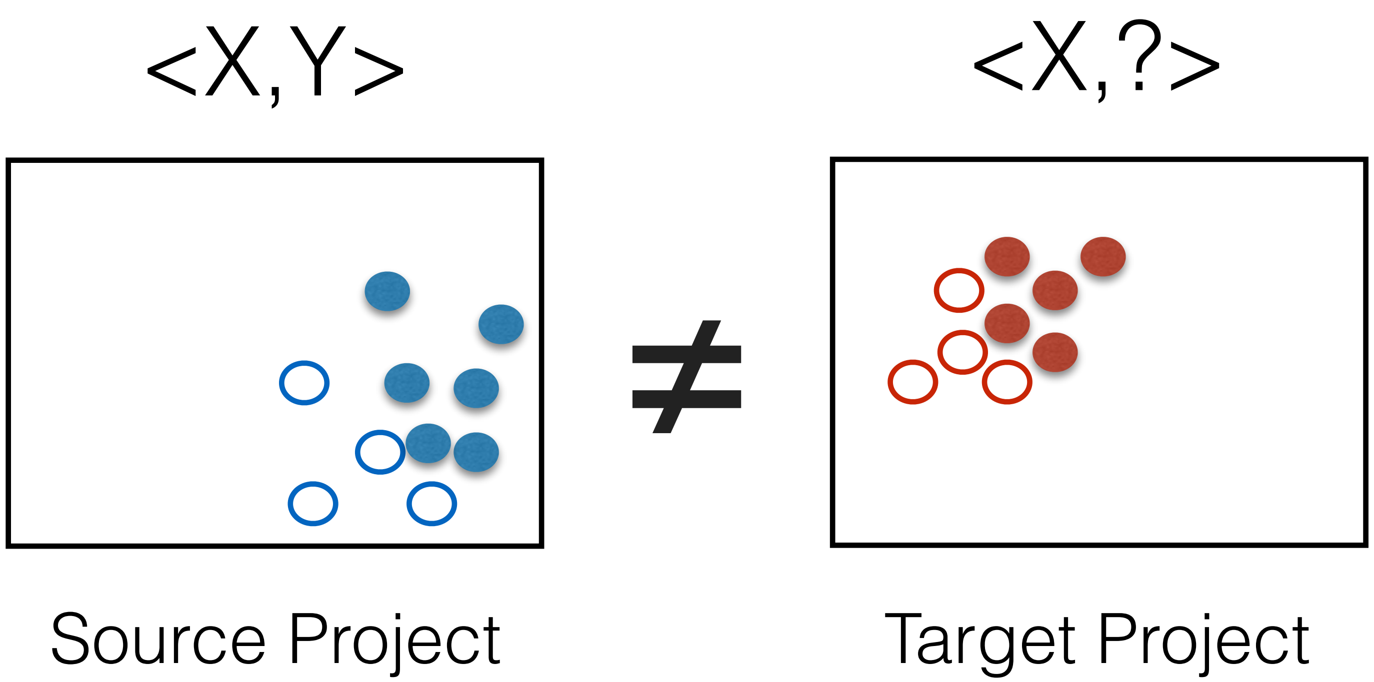
Multi-Source Data



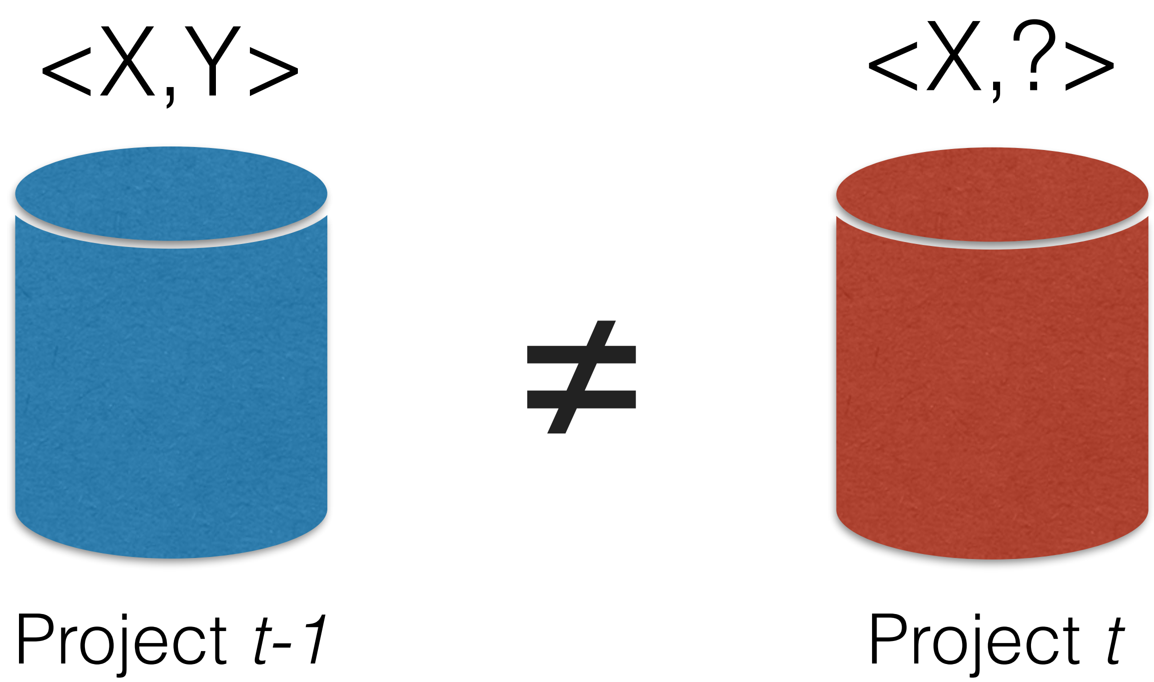
Multi-Source Data



Multi-Source Data



Temporal Data



L. Minku and X. Yao. "How to Make Best Use of Cross-company Data in Software Effort Estimation?", ICSE 2014.

Typical Setback

Ignore the potentially different data distributions, by adopting techniques not prepared for such differences.

- Models initially perform well, but then become poor.
- Models perform poorly straight away.

Example: Software Defect Prediction

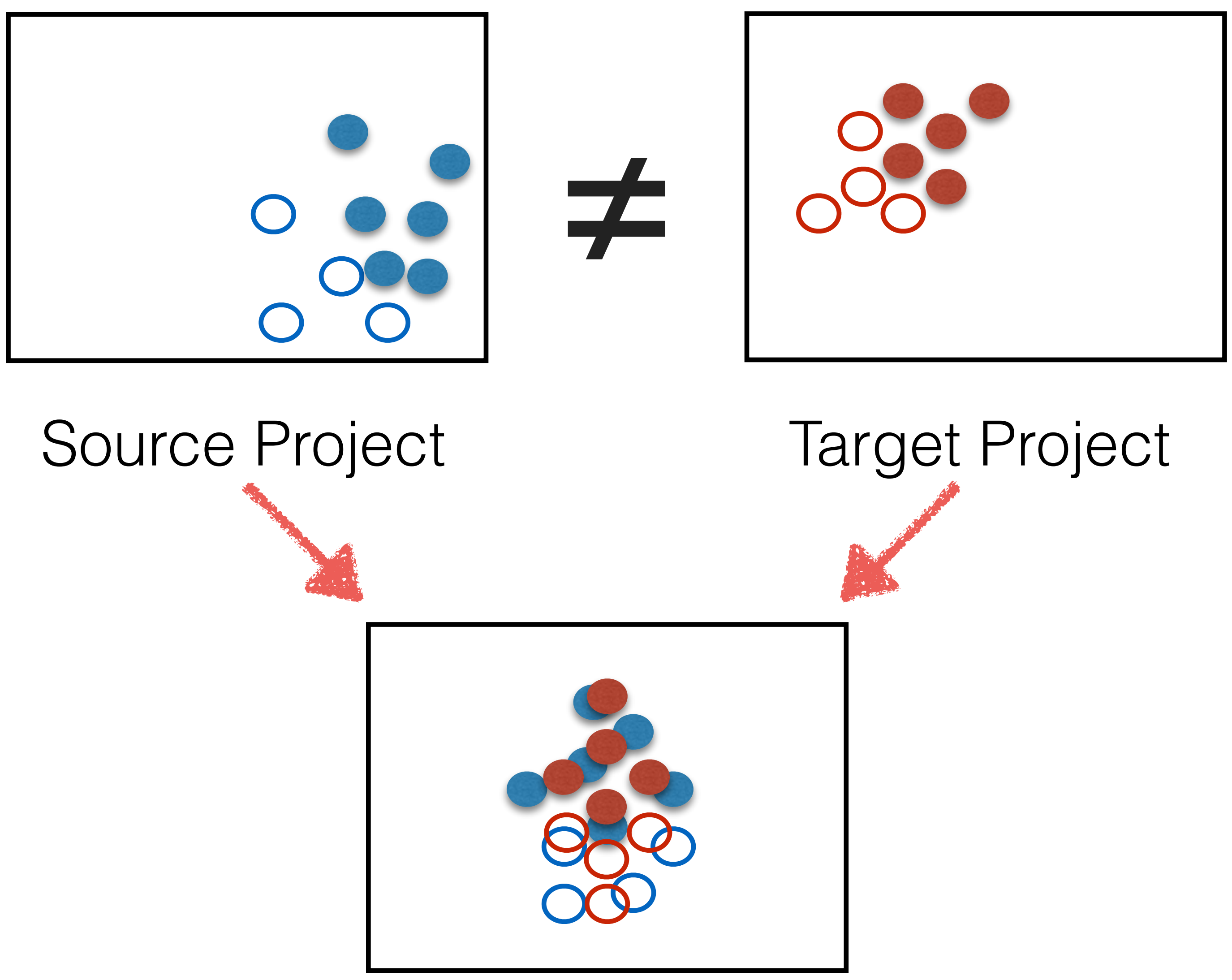
Out of 622 source-target project combinations, only 21 had precision, recall, and accuracy larger than 75%; a success rate of 3.4%.

T. Zimmermann, M. Nagappan, N. Gall, E. Giger, B. Murphy. Cross-project defect prediction: a large scale experiment on data vs. domain vs. process, FSE 2009.

Avoiding Setback

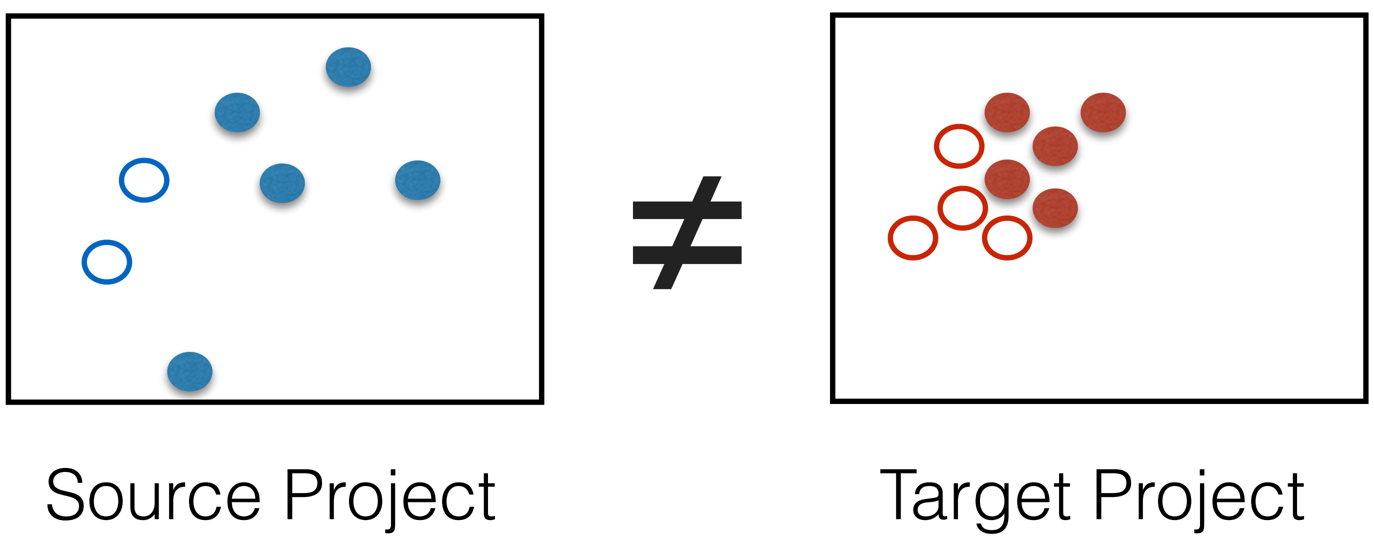
Adopt approaches that are prepared for multi-source or temporal data.

Transfer Learning — Data Transformation



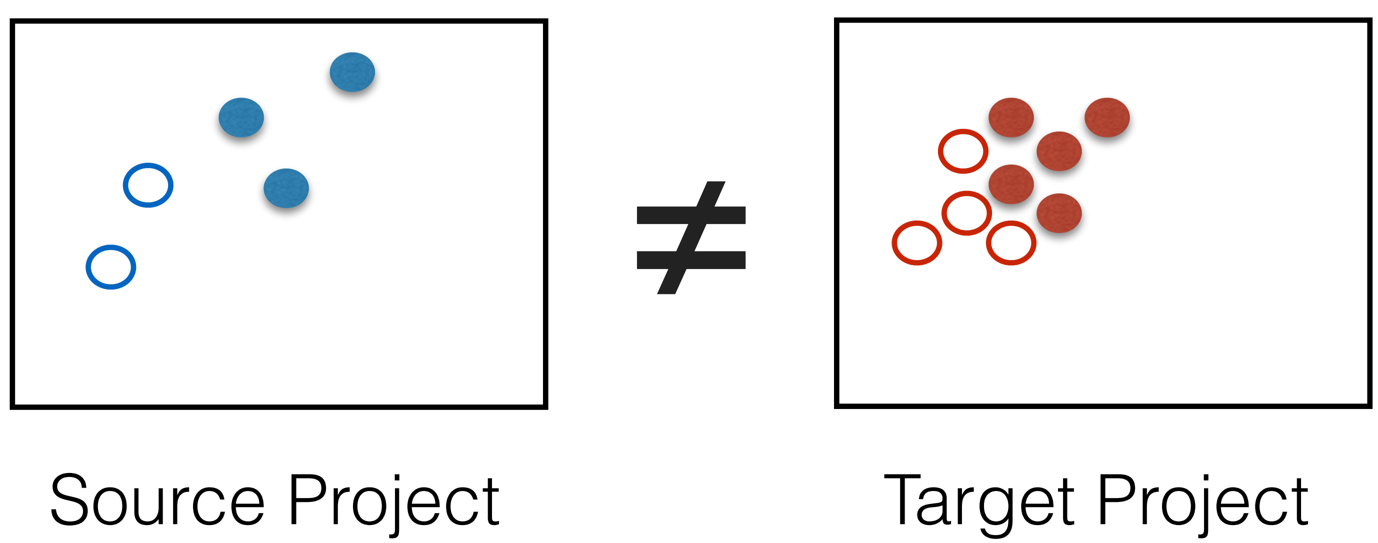
J. Nam, S.J. Pan and S. Kim. Transfer Defect Learning, ICSE 2013.

Transfer Learning — Filtering



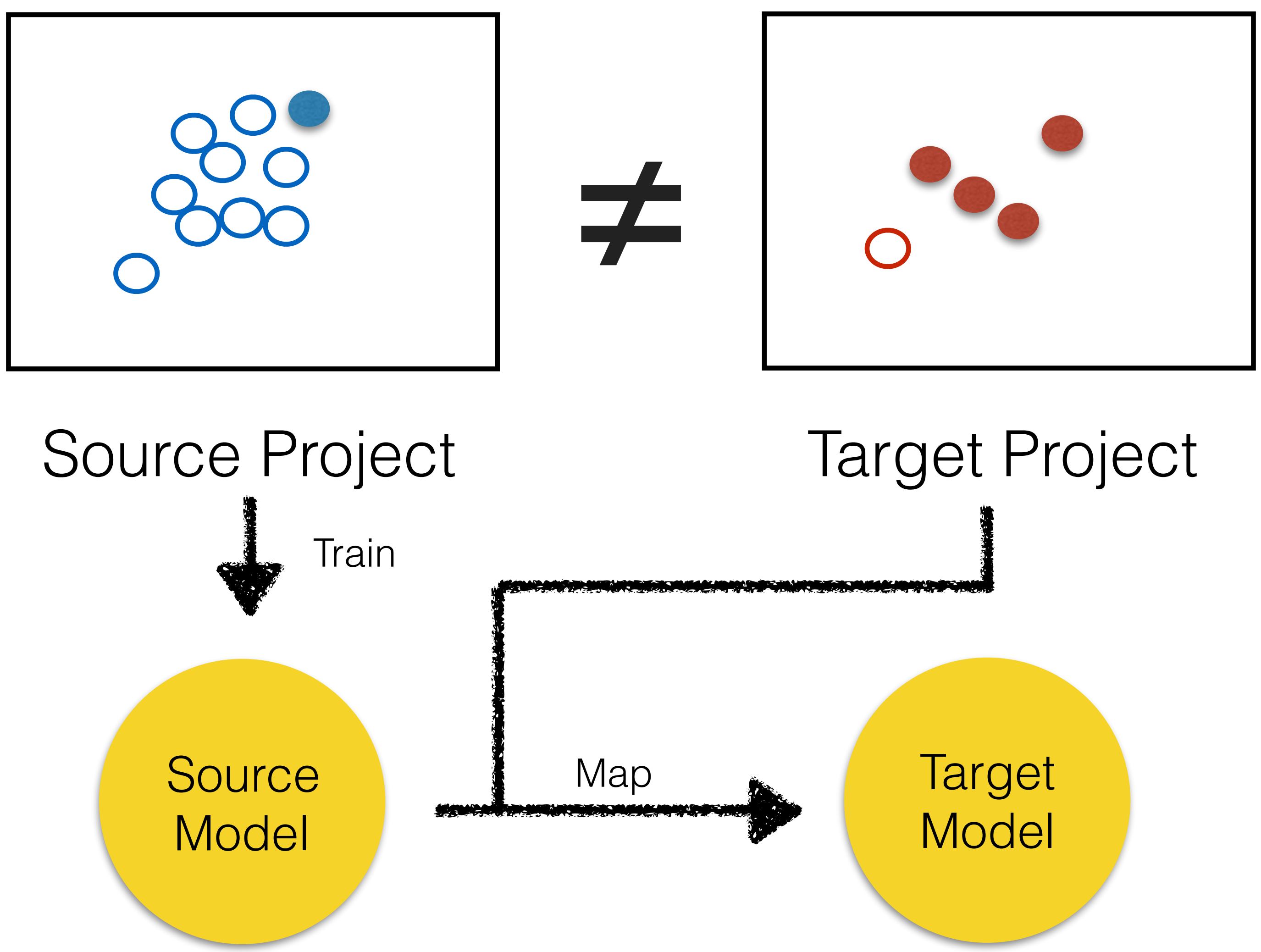
B. Turhan, T. Menzies, A. Bener, J. Distefano. “On the Relative Value of Cross-Company and Within-Company Data for Defect Prediction”, EMSE 2009.

Transfer Learning — Filtering



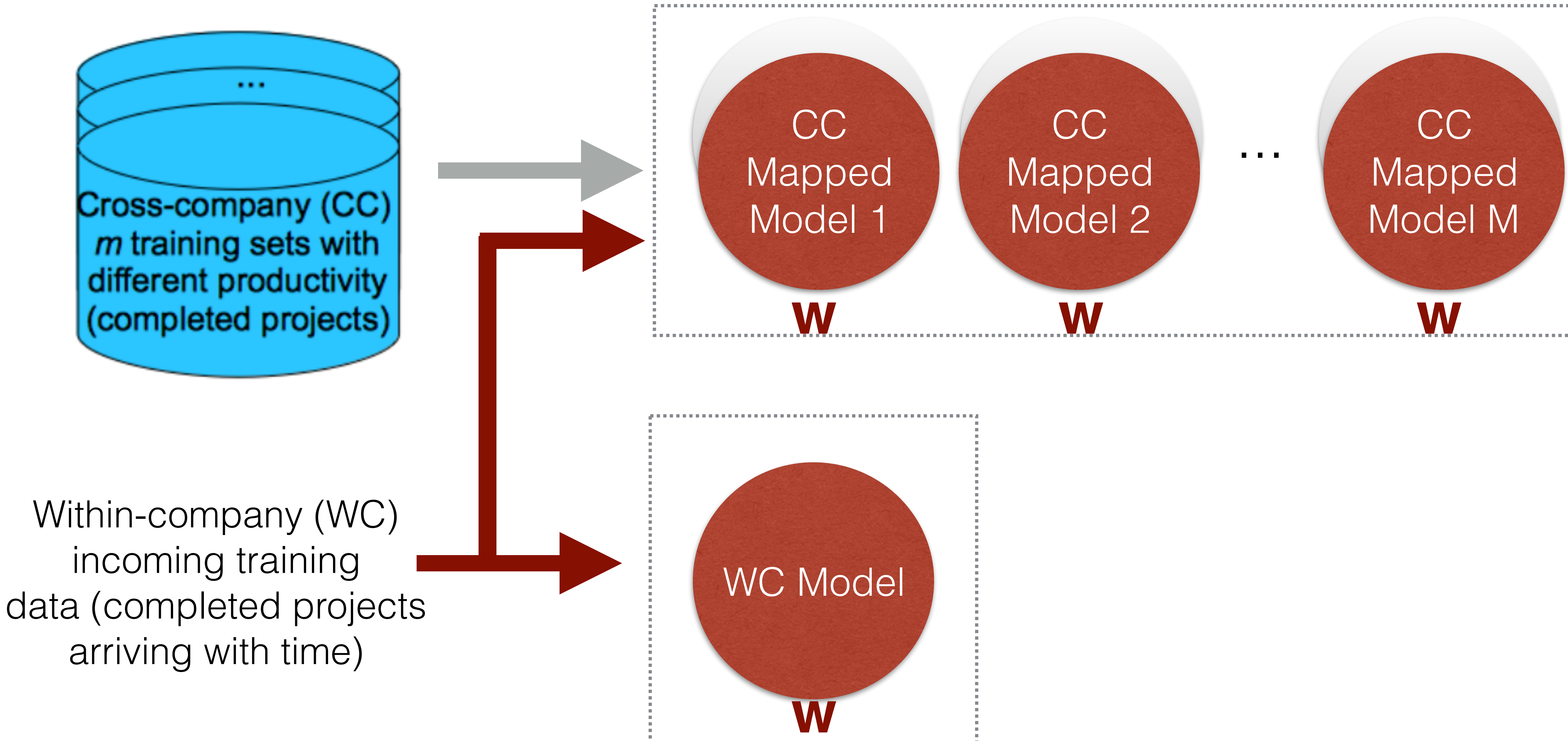
B. Turhan, T. Menzies, A. Bener, J. Distefano. “On the Relative Value of Cross-Company and Within-Company Data for Defect Prediction”, EMSE 2009.

Transfer Learning — Mapping Predictions



L. Minku and X. Yao. "How to Make Best Use of Cross-company Data in Software Effort Estimation?", ICSE 2014.

Machine Learning for Non-Stationary Environments



L. Minku and X. Yao. "How to Make Best Use of Cross-company Data in Software Effort Estimation?", ICSE 2014.

Example: Software Effort Estimation

Database	Approach	MAE
KitchenMax	RT	2441.0241
	Dycom-RT	2208.6522
	P-value	3.82E-11
CocNasaCoc81	RT	319.4572
	Dycom-RT	161.7917
	P-value	4.04E-06
ISBSG2000	RT	2753.3726
	Dycom-RT	2494.6639
	P-value	4.72E-02
ISBSG2001	RT	3621.9598
	Dycom-RT	2543.9495
	P-value	3.21E-06
ISBSG	RT	3253.9349
	Dycom-RT	3122.6603
	P-value	5.56E-02

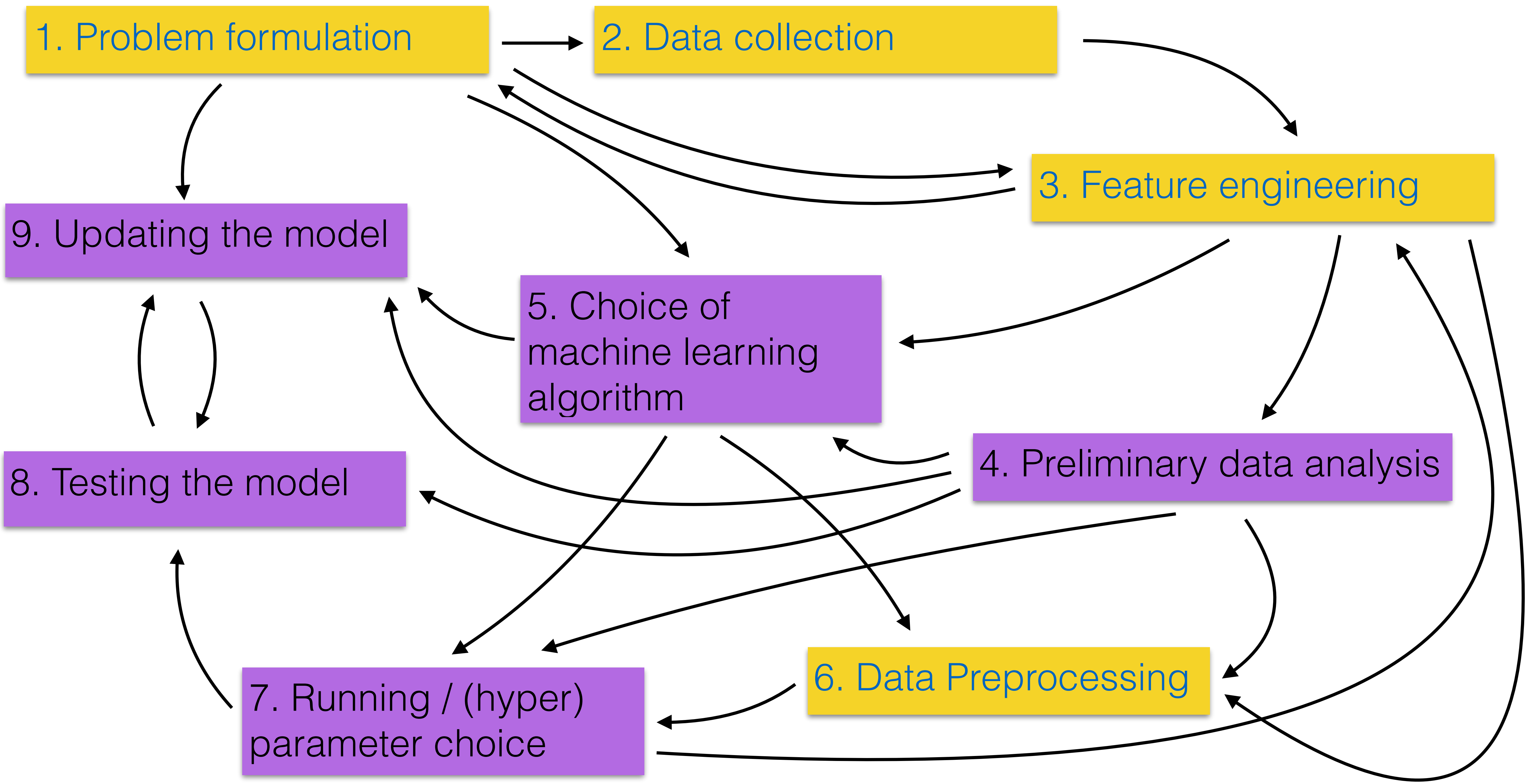
Dycom managed to obtain reduce the need for target training examples by 90%!

L. Minku and X. Yao. "How to Make Best Use of Cross-company Data in Software Effort Estimation?", ICSE 2014.

Four Considerations

- Problem relevance.
- Multi-source and temporal data.
- **Class imbalance.**
- Parameter tuning.

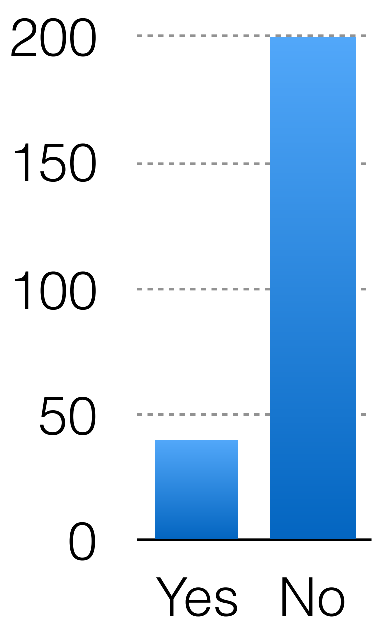
Data Science Involves Several Interdependent Steps



Preliminary Data Analysis

LOC	Halstead difficulty	Expertise of majority of	Defective?
{1,2,…}	{1,2,…}	{L, M, H}	{Yes, No}

Class imbalance



Example of Class Imbalance in Software Defect Prediction

TABLE I
PROMISE DATA SETS, SORTED IN ORDER OF THE IMBALANCE RATE (DEFECT%: THE PERCENTAGE OF DEFECTIVE MODULES)

data	language	examples	attributes	defect%
mc2	C++	161	39	32.29
kc2	C++	522	21	20.49
jm1	C	10885	21	19.35
kc1	C++	2109	21	15.45
pc4	C	1458	37	12.20
pc3	C	1563	37	10.23
cm1	C	498	21	9.83
kc3	Java	458	39	9.38
mw1	C	403	37	7.69
pc1	C	1109	21	6.94

Source: S. Wang and X. Yao. Using Class Imbalance Learning for Software Defect Prediction, IEEE TR 62(2):434-443

Typical Setback

Use inadequate performance metrics to evaluate the predictive models.
You may think the approach works well when in fact it doesn't!

TABLE I
PROMISE DATA SETS, SORTED IN ORDER OF THE IMBALANCE RATE (DEFECT%: THE PERCENTAGE OF DEFECTIVE MODULES)

data	language	examples	attributes	defect%
mc2	C++	161	39	32.29
kc2	C++	522	21	20.49
jm1	C	10885	21	19.35
kc1	C++	2109	21	15.45
pc4	C	1458	37	12.20
pc3	C	1563	37	10.23
cm1	C	498	21	9.83
kc3	Java	458	39	9.38
mw1	C	403	37	7.69
pc1	C	1109	21	6.94

E.g., an algorithm predicting all examples as non-defective would have 93.06% accuracy.

Source: S. Wang and X. Yao. Using Class Imbalance Learning for Software Defect Prediction, IEEE TR 62(2):434-443

Avoiding Setback

Adopt performance metrics appropriate for class imbalance.

Evaluating Classifiers for Class Imbalanced Data

- Accuracy is inadequate.
  - *(TP + TN) / (P + N)*
- Recall on each class separately is not sensitive to the imbalance status.
  - *TP / P* and *TN / N*.
- G-mean is not sensitive to the imbalance status.
  - $\sqrt{TP/P * TN/N}$
- ROC Curve is adequate.
  - Recall on positive class *(TP / P)* vs False Alarms *(FP / N)*

Related Setback

Adoption of inadequate machine learning algorithm.

- Most machine learning algorithms give the same importance to each separate training example.
- This can result in poor predictive performance for class imbalanced problems.

Avoiding Setback

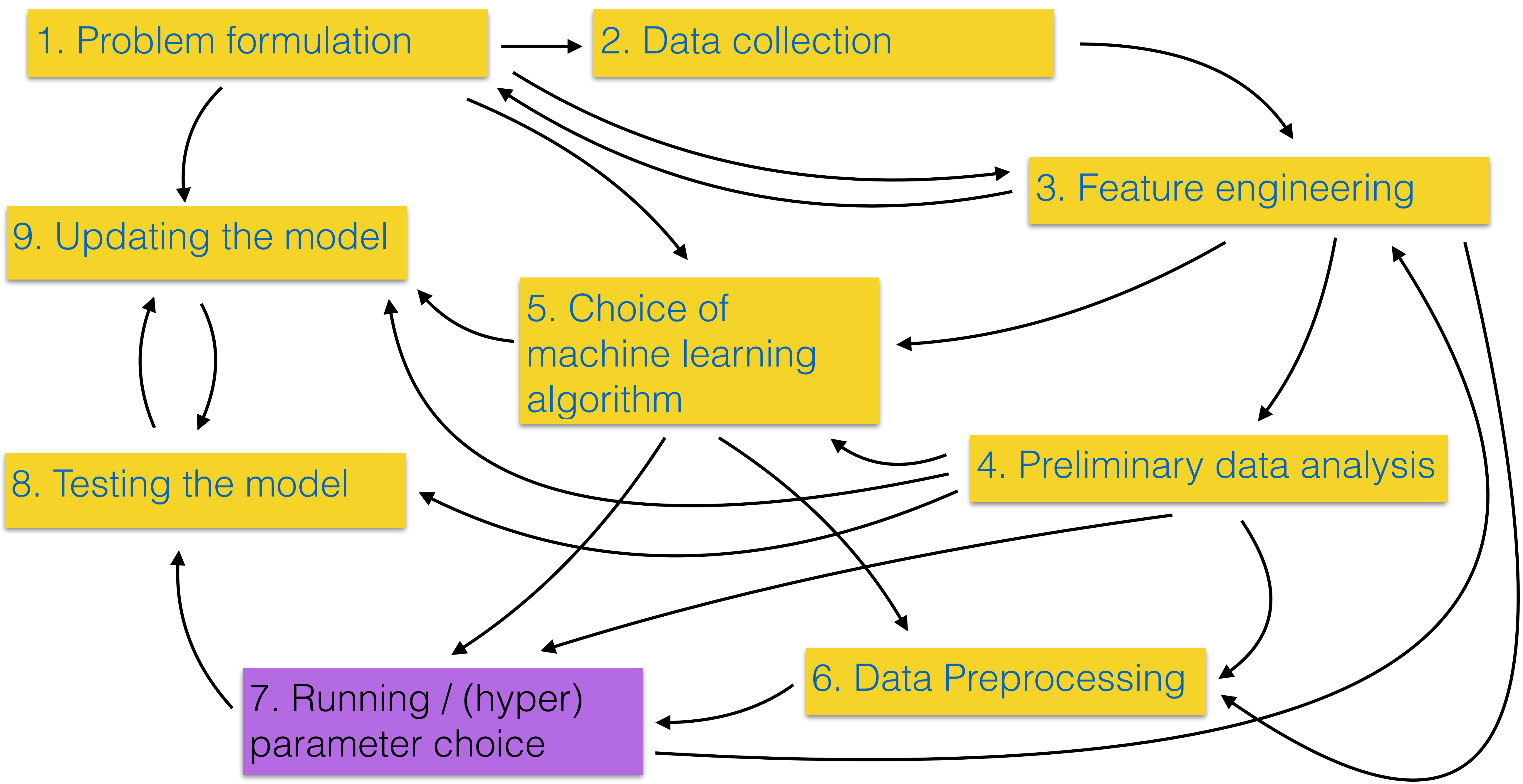
Adopt resampling strategies or cost-sensitive machine learning algorithms.

S. Wang and X. Yao. Using Class Imbalance Learning for Software Defect Prediction, IEEE TR 2013.

Four Considerations

- Problem relevance.
- Multi-source and temporal data.
- Class imbalance.
- **Parameter tuning.**

Data Science Involves Several Interdependent Steps



The Impact of (Hyper)Parameters

Example: Multilayer Perceptron for Software Effort Estimation

MAE across time steps		Kitchenham	Maxwell	SingleISBSG
Best PS	MAE	2046.35	5358.02	2754.78
	std.	2868.96	1979.71	1006.01
Default PS	MAE	2474.78	7893.26	3682.47
	std.	2846.06	3629.54	1254.03
Worst PS	MAE	7.42E+138	1.19E+155	1.07E+153
	std.	4.71E+140	Inf	Inf

Cohen's d effect size and Wilcoxon Sign-Rank's p-values

Effect Size	Kitchenham	Maxwell	SingleISBSG
best vs. worst	2.5863E+135 (6.77E-21)+	6.011E+151 (6.15E-10)+	1.0636E+150 (3.51E-11)+
best vs. default	0.149 (2.66E-22)+	1.281 (6.15E-10)+	0.922 (3.51E-11)+

L. Song, L. Minku, X. Yao. The Impact of Parameter Tuning on Software Effort Estimation Using Learning Machines, PROMISE 2013.

The Impact of (Hyper)Parameters

What are good values?

Best values are data-dependent.

Typical Setback

Use of default (hyper)parameter values, or values that did well for other data.

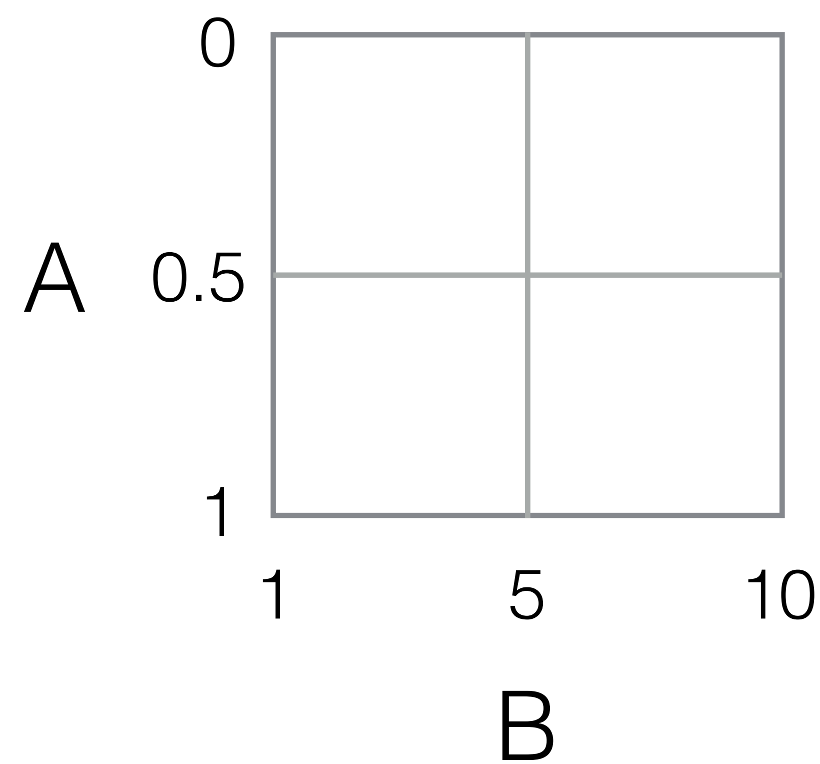
Avoiding Setback

Tune (hyper)parameters for the data in hands.

Tuning (Hyper)Parameter Values

Grid search: investigate all combinations of a pre-defined set of values.

- Cross-validation
- Leave-one-out cross-validation
- Repeated Holdout
- Out-of-sample bootstrap



Tantithamthavorn, C., McIntosh, S., Hassan, A.E., Matsumoto, K. Automated parameter optimization of classification techniques for defect prediction models, ICSE 2016.

Tantithamthavorn, C., McIntosh, S., Hassan, A.E., Matsumoto, K. An empirical comparison of model validation techniques for defect prediction models, IEEE TSE 2017.

Tuning (Hyper)Parameter Values

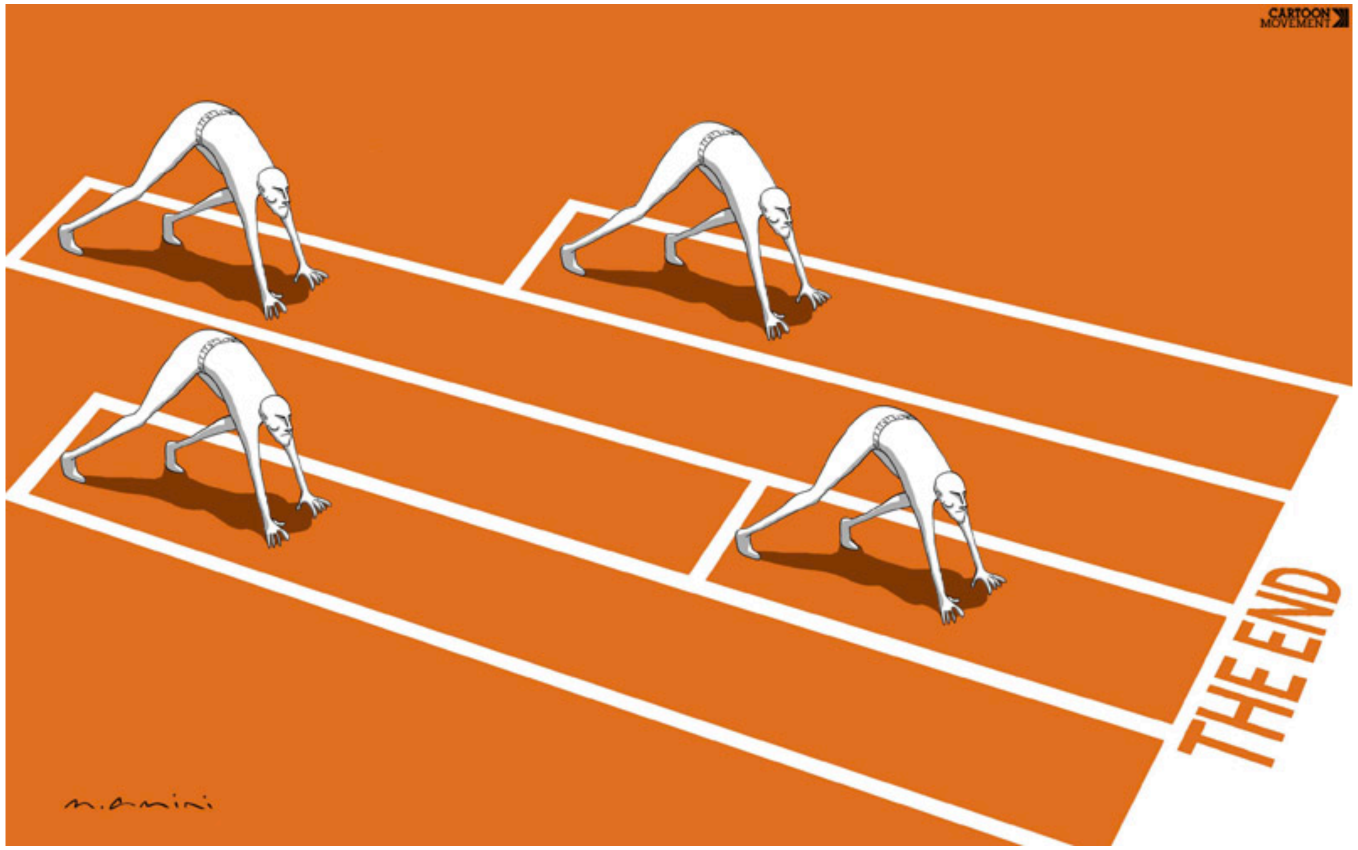
- Automated tuning: does not require to specify specific values to try out, only the ranges of each hyperparameter.
    - E.g.: differential evolution.

W. Fu, T. Menzies. Easy over hard: a case study on deep learning, FSE 2017.

A. Agrawal, T. Menzies. Is “better” data better than “better” data miners"? ICSE 2018.

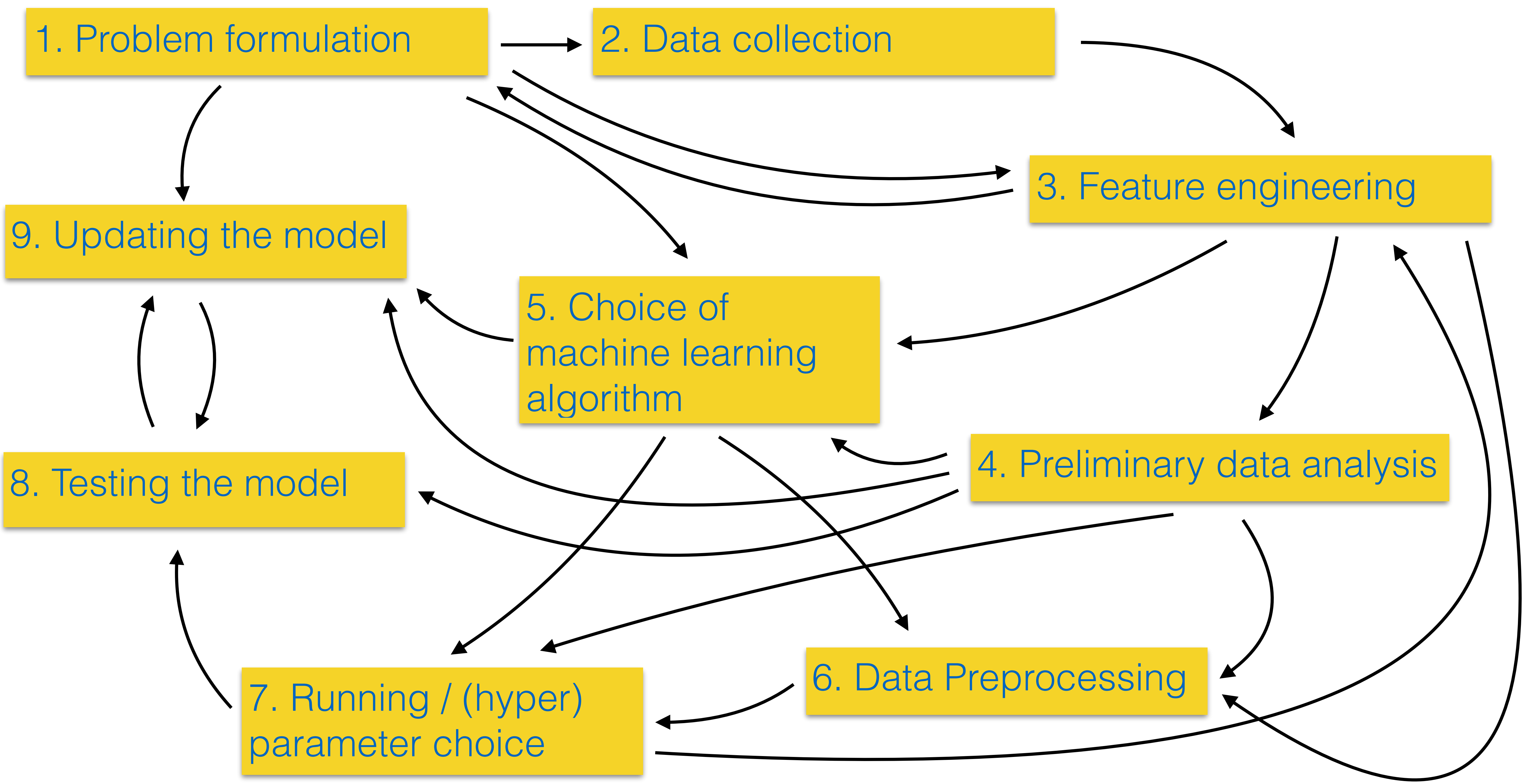
Related Setback

Uneven parameter tuning, leading to unfair comparisons and wrong conclusions.



PS: depending on the purpose of the experiment, the use of default parameters is ok. However, it needs to be very well justified.

Conclusions



Conclusions

Four important considerations:

- Problem relevance.
- Multi-source and temporal data.
- Class imbalance.
- Parameter tuning.

Conclusions

Overlooking these considerations may lead to:

- Useless problem.
- Poor performing predictive models.
- Wrong conclusions.